TRAFFIC OPTIMIZATION BETWEEN CITIES

OPTIMIZATION REPORT

Realized by: BOUJDID Abderrahim

Under direction: Ms. EL MKHALET Mouna

----------Summary----------

Introduction

The traveling salesman problem

Definitions

Algorithms

- *Dijkstra's Algorithm*
- *The Ant Colony Optimization algorithm*

Perspective and applications

Conclusions

References

I. Introduction:

Transport is one of the fields that civil engineers are interest with, and its development. As a matter of fact, it is considered one of the most important sectors of economic in the whole world, and by making it faster and smarter it will participate in the development of human civilization and will open another whole world of possibilities and challenges. Beside that currently transport is the 2nd big source of pollution after electric production. Then by optimization transport of goods and persons we will serve the big two goals of human beans.

Therefore, in this paper we will discover some algorithms that provide path optimization and discover it applications.

Clarification: I have detailed algorithm explication because I used an example of how to apply this algorithm (Dijkstra essentially) in real world if u want to optimize a road while serving as much as possible of points.

II. The traveling salesman problem:

The traveling salesman problem is a classical problem in combinatorial optimization. The task is to find the shortest possible tour through a set of N cities that passes through each city exactly once. This problem is known to be NP-complete (**nondeterministic polynomial)**, and it is generally believed that the computational power needed to solve it grows exponentially with the number of cities. The time is in O (n! *n).

Example:

If we have 16 cities then there are 334 764 638 208 000 distinct routes are possible. And to compile it in computer it will take at least 21 days of continues work.

And just for 20 cities there are 48 658 040 163 532 800 000 distinct routes are possible. And to compile it in computer it will take 514,31 years of continues work.

And if you want to reduce this time, either you have a super-fast computer such as quantum computer – and that is not an option in present but it is very close to be - or to find a much smarter way **"Algorithm"** to be able to find the short road to serve several areas, in a reasonable time.

III. Definitions:

Optimization: is the selection of a best element, with regard to some criterion, from some set of available alternatives. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries.

Algorithm: is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation. Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, and other tasks.

IV. Algorithms:

1. Dijkstra's Algorithm

Named after its inventor Edsger Dijkstra (1930-2002) Truly one of the "founders" of computer science; this is just one of his many contributions

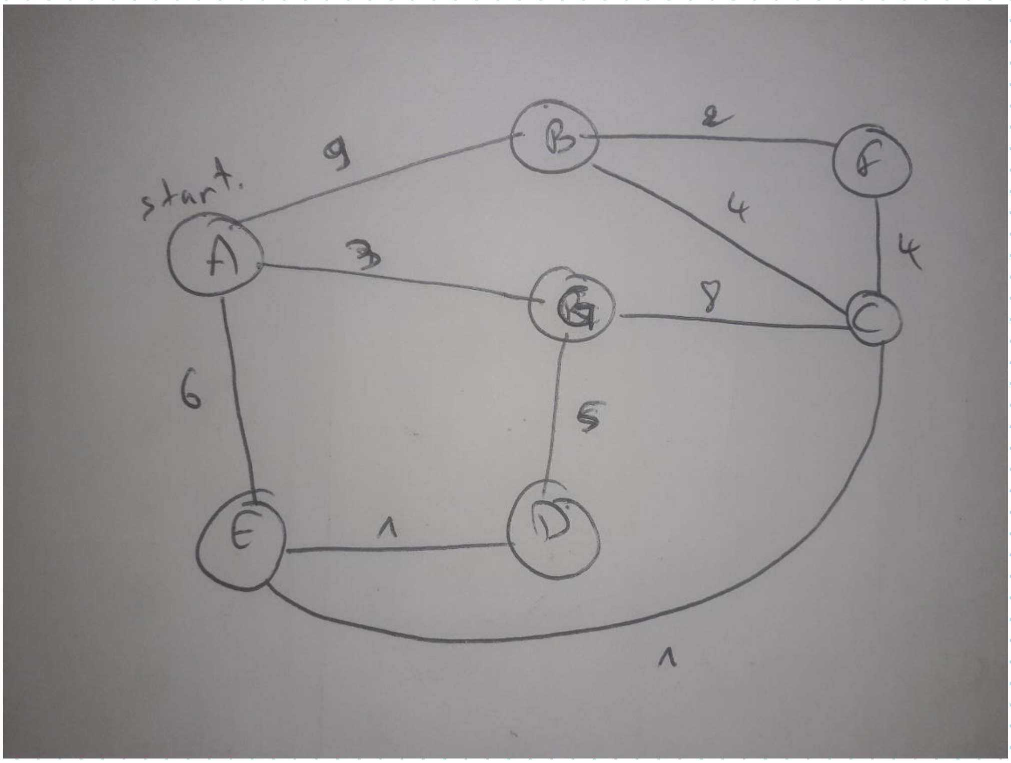
Goal: From one starting vertex (city, point….), watt are the shortest paths to each of the other vertices (for a weighted graph)?

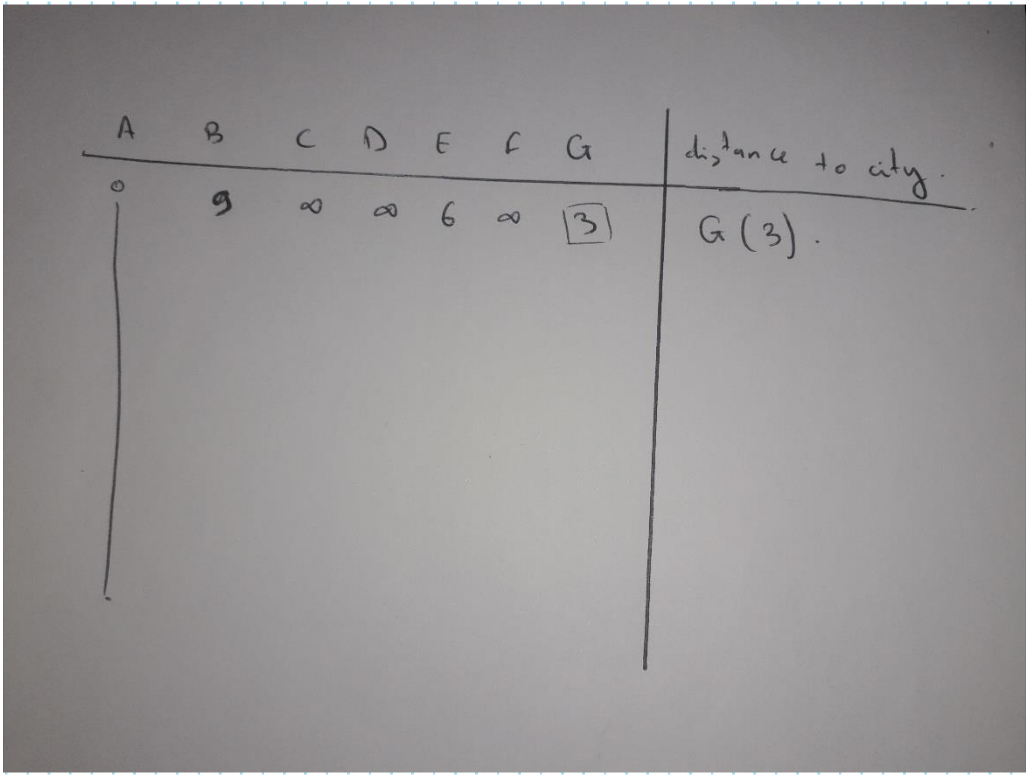
Idea: Similar to BFS (*Breadth-first search-Algorithme de parcours en largeur-)*

- Repeatedly increase a "set of vertices with known shortest distances".

- Any vertex not in this set will have a “best distance so far”.
- Each vertex has a “cost” to represent these shortest/best distances.
- Update costs (i.e., “best distances so far”) as we add vertices to set.

To apply these steps easily we us a table as shown in the example bellow:



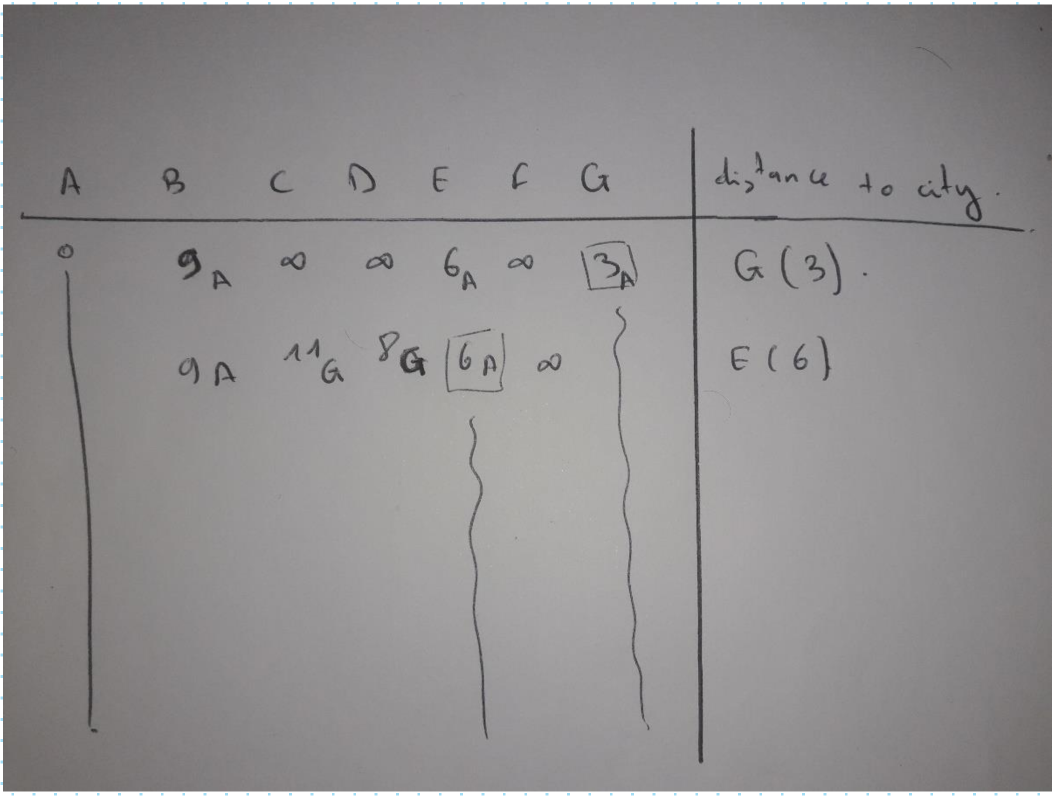


A	B	C	D	E	F	G	distance to city.
0	9	∞	∞	6	∞	$\boxed{3}$	G (3).

This table is used to apply the Dijkstra algorithm by setting in a chosen city X and calculate the distance to the cities that have link to X.

In this table above we are placed in A and there is only link to B, E, and G by these distances (9, 6, 3)

Then we choose the nearest city (in our case it's G by 3) and get into it and star calculating **the total distance** from A but without return to A.

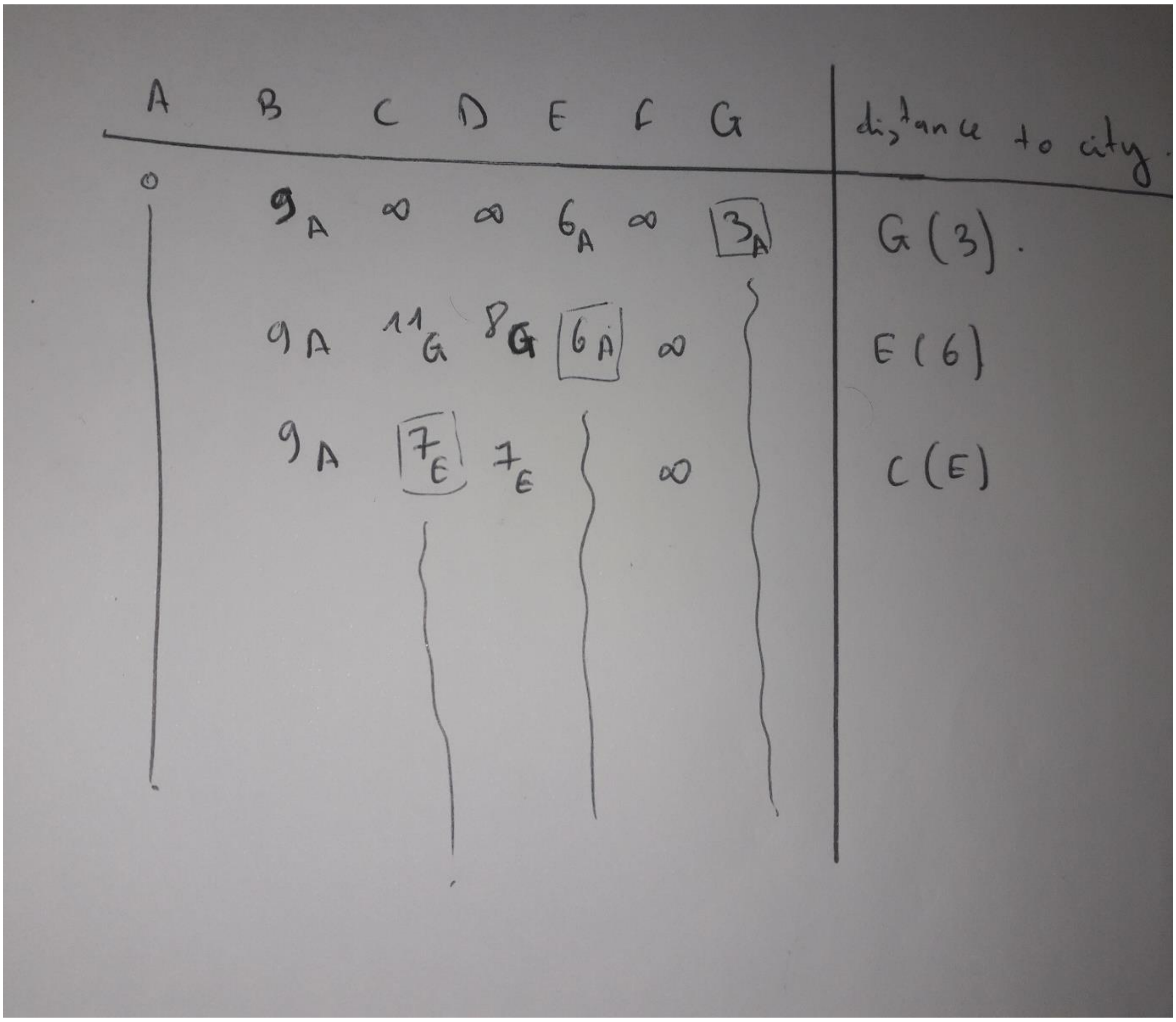


A	B	C	D	E	F	G	distance to city.
0	9_A	∞	∞	6_A	∞	$\boxed{3_A}$	G (3).
	9_A	11_G	8_G	$\boxed{6_A}$	∞		E (6)

In this step we are in G and there is a link to (C, D) by distance (8,5) and by adding A-G=3 then it is (11,8). The other cities kept their values. Next step we will be in city E. Again, we star calculating **the total distance** from A but without return to A or G.

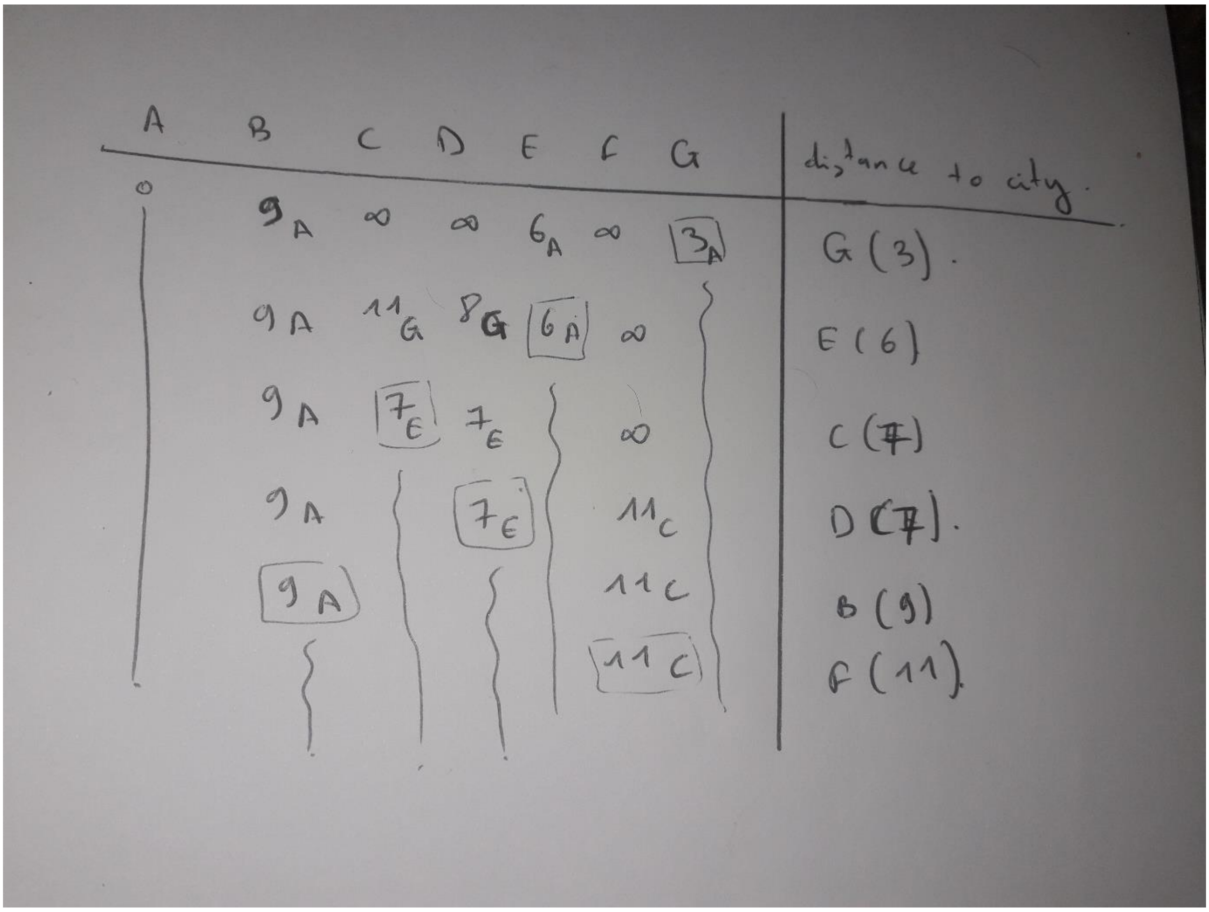
The city E have link to (D, C) (1,1), by adding A-E=6 it become (7,7). And we have already A-C=9 A-D=8, so we choose the smallest distance and we put it in our table.

For next step in our case, we have two equal minimal distances (7), so we can choose whatever we want, so we choose C next step.



A	B	C	D	E	F	G	distance to city.
0	9_A	∞	∞	6_A	∞	$\boxed{3_A}$	G(3).
	9_A	11_G	8_G	$\boxed{6_A}$	∞		E(6)
	9_A	$\boxed{7_E}$	7_E		∞		C(E)

And the final table will be:



A	B	C	D	E	F	G	distance to city.
0	9_A	∞	∞	6_A	∞	$\boxed{3_A}$	G (3).
	9_A	11_G	8_G	$\boxed{6_A}$	∞		E (6)
	9_A	$\boxed{7_E}$	7_E		∞		C (7)
	9_A		$\boxed{7_E}$		11_C		D (7).
	$\boxed{9_A}$				11_C		B (9)
					$\boxed{11_C}$		F (11)

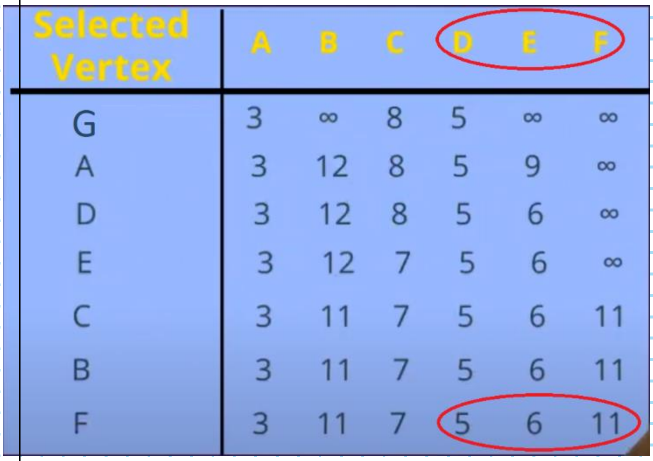
Finally, the minimal distance from A to (B, C, D, E, F, G) is (9,7,7,6,11,3)

Until now we just calculate the minimal distance from a starting vertex to the each of the other vertex, not the optimum distance to serve several points.

In order to achieve our goal, we have run this processor to each of the cities to serve and remake a simplified graph containing only the wanted cities to serve.

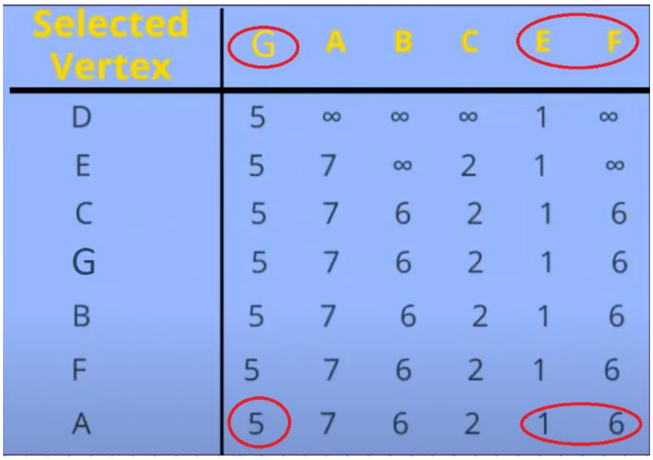
For this example, let's assume that we want to serve (D, E, F, G) knowing that we start at G.

first, we have to do for each city its table, then we will look for the minimum distance between these cities, and make our simplified graph.



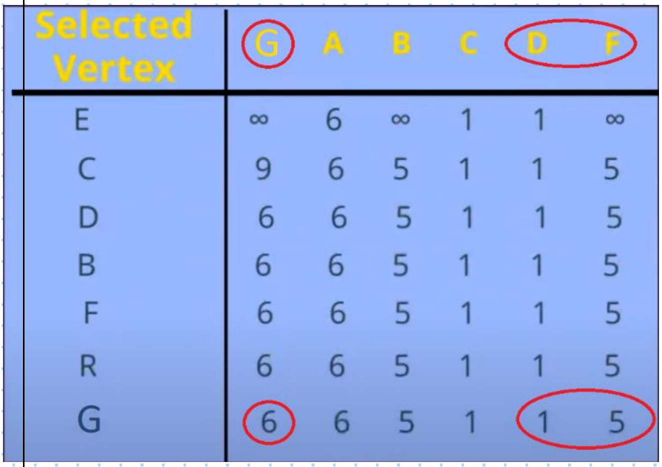
Selected Vertex	A	B	C	D	E	F
G	3	∞	8	5	∞	∞
A	3	12	8	5	9	∞
D	3	12	8	5	6	∞
E	3	12	7	5	6	∞
C	3	11	7	5	6	11
B	3	11	7	5	6	11
F	3	11	7	5	6	11

Start city G



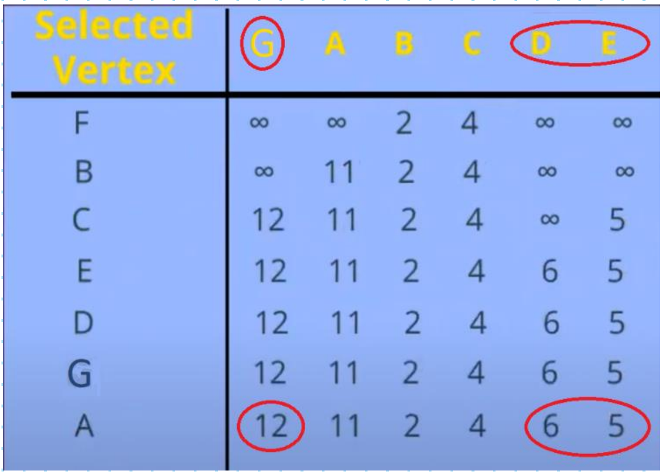
Selected Vertex	G	A	B	C	E	F
D	5	∞	∞	∞	1	∞
E	5	7	∞	2	1	∞
C	5	7	6	2	1	6
G	5	7	6	2	1	6
B	5	7	6	2	1	6
F	5	7	6	2	1	6
A	5	7	6	2	1	6

Start city D



Selected Vertex	G	A	B	C	D	F
E	∞	6	∞	1	1	∞
C	9	6	5	1	1	5
D	6	6	5	1	1	5
B	6	6	5	1	1	5
F	6	6	5	1	1	5
R	6	6	5	1	1	5
G	6	6	5	1	1	5

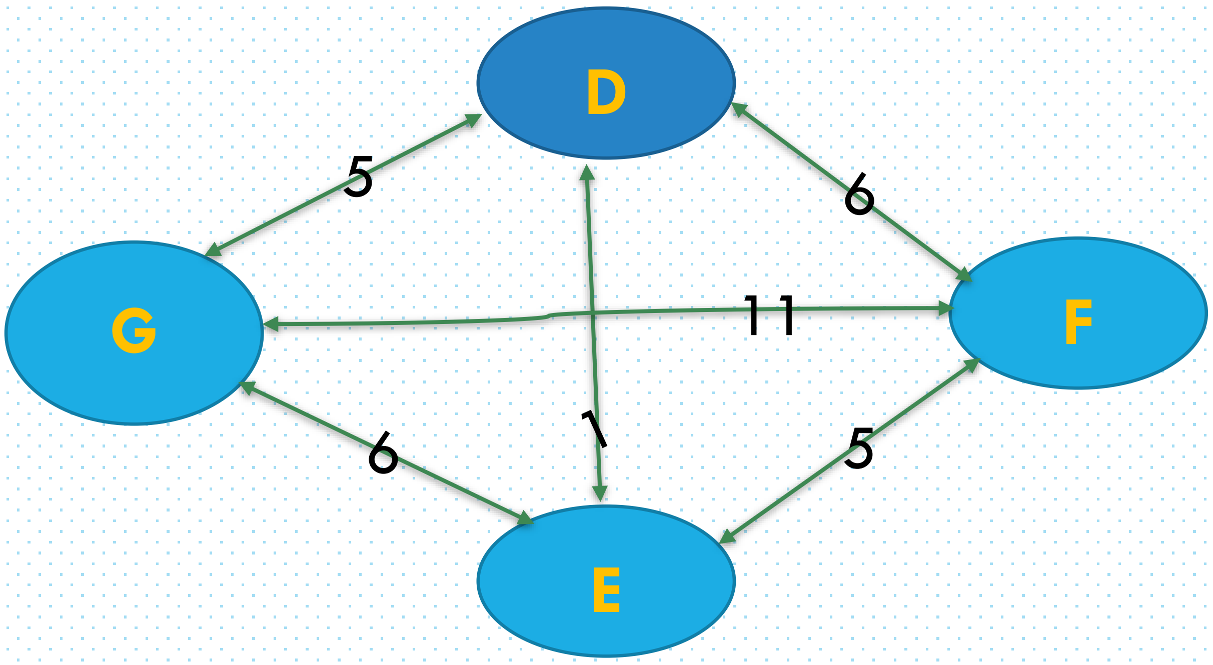
Start city E



Selected Vertex	G	A	B	C	D	E
F	∞	∞	2	4	∞	∞
B	∞	11	2	4	∞	∞
C	12	11	2	4	∞	5
E	12	11	2	4	6	5
D	12	11	2	4	6	5
G	12	11	2	4	6	5
A	12	11	2	4	6	5

Start city F

After we make this tables, we now have to draw the simplified graph using data from the tables above.



Finlay with this diagram we can deduce the minimal distance from a starting point choosing the smallest distance

In our case its G→D→E→F→G with 22 total distance

→Conclusions:

this algorithm helps to simplify the problem to a much easy one and then apply a simple algorithm by choosing the smallest distance.

This algorithm is of polynomial complexity. More precisely, for n vertices and an arc, the time is in O ((a + n) log n).

Although, with all of this characteristic this algorithm can't find the ultimate solution for the problems for much higher number of cities.

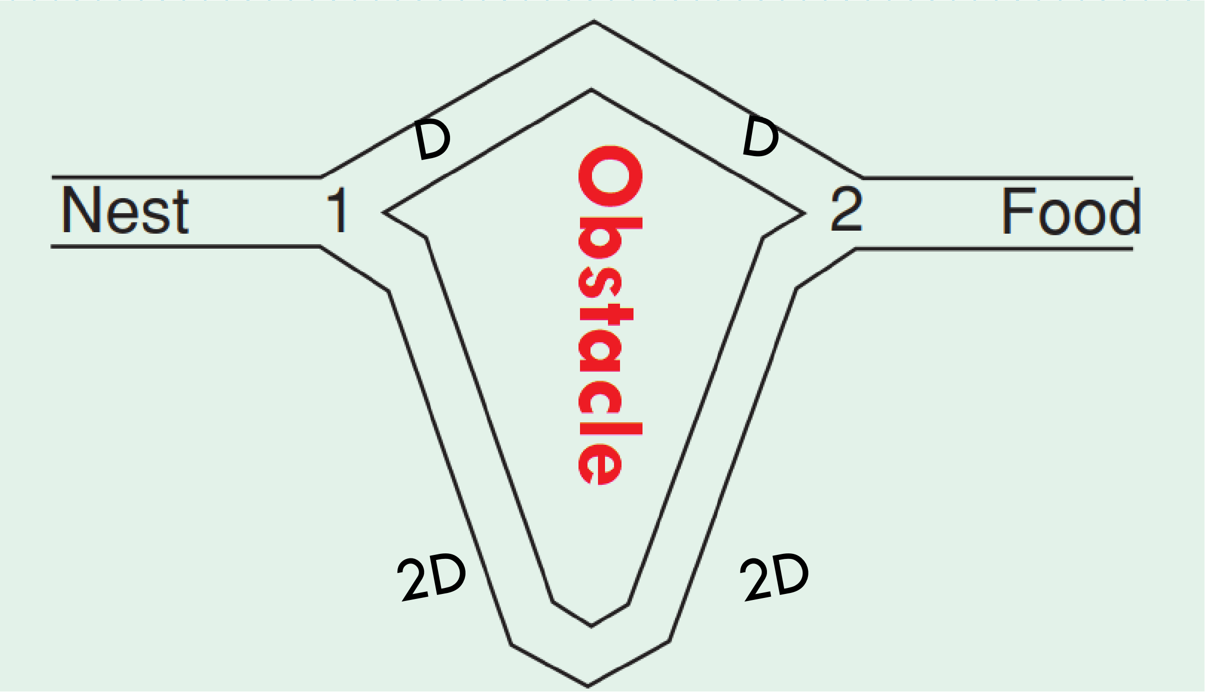
2. The Ant Colony Optimization algorithm:

Ant colony optimization has been formalized into a metaheuristic for combinatorial optimization problems by Dorigo and co-workers

So, in order to understand this algorithm, we will go throw this example:

In many ant species, ants walking to and from a food source deposit on the ground a substance called **pheromone**. Other ants perceive the presence of pheromone and tend to follow paths where pheromone concentration is higher. Through this mechanism, ants are able to transport food to their nest in a remarkably effective way.

So, in the first time the ants are walking in a strict line then we pose an obstacle in their way. This obstacle creates two possible ways, but ants cannot see the end of any one, and there is no pheromone in any road (there is no ant that passe from any way before, therefore the possibility of an ant to choose one of a way is equal for both of them 50%



Let's assume that there are 10 ants in each side of the obstacle and each ant can travel 1D in a second.

After two seconds 5 ants would pass from 1→2 and another 5 from 2→1 using the up way. So, the pheromone will be equal to 2*5 =10

For the down way, the ants in each side would be in the top of the way and therefore the pheromone = 5

And then for the next group ants there is 66.66% that they will choose the up way and 33.33% will choose the down way.

And after a while it will be 100% of the ants going by the up way

a) Modulization:

Ant System is the first ACO algorithm proposed in the literature. Its main characteristic is that, at each iteration, the pheromone values are updated by all the m ants that have built a solution in the iteration itself. The pheromone **τij**, associated with the edge joining cities i and j, is updated as follows:

$$\tau ij \leftarrow (1 - \rho) \cdot \tau ij + \sum_{k=1}^{m} \ \Delta \tau ij^{k}$$

where ρ is the evaporation rate, m is the number of ants, and τ k i j is the quantity of pheromone laid on edge (i, j) by ant k:

$$\Delta \tau ij^{k} \ = \begin{cases} Q/L_{k} \quad \text{if ant k used edge } (i, j)\text{in its tour,} \\ \qquad\qquad\quad 0 \text{ otherwise,} \end{cases}$$

where Q is a constant, and L k is the length of the tour constructed by ant k.

In the construction of a solution, ants select the following city to be visited through a stochastic mechanism. When ant k is in city i and has so far constructed the partial solution s p, the probability of going to city j is given by:

$$pij^{k} \ = \begin{cases} \dfrac{\Delta \tau ij^{\alpha} . \eta ij^{\beta}}{\sum_{cil \in N(s^{p})} \ \tau il^{\alpha} . \eta il^{\beta}} \\ \quad 0 \quad \text{otherwise} \end{cases}$$

where N(s p) is the set of feasible components; that is, edges (i, l) where l is a city not yet visited by the ant k. The parameters α and β control the relative importance of the pheromone versus the heuristic information η i j, which is given by:

$$\eta ij = \frac{1}{dij}$$

where dij is the distance between cities i and j.

→Conclusions:

Nowadays hundreds of researchers worldwide are applying ACO to classic N P-hard optimization problems, while only a few works concern variations that include dynamic and stochastic aspects as well as multiple objectives. The study of how best to apply ACO to such variations will certainly be one of the major research directions in the near future

V. Perspective and applications:

These algorithms are very helpful in many domains such as communication by optic fibers to select the shortest way to decrees speed connection for users, and to calculate for instance to establish tracks of electricity lines or oil pipelines. The algorithms have also been used to calculate optimal long-distance footpaths in Ethiopia and contrast them with the situation on the ground.

Location-routing problems arise in a variety of practical contexts where it is required to simultaneously locate facilities and design vehicle routes through these facilities and a number of other sites. For example, Watson-Gandy and Dohrn solve the problem of locating depots and of constructing vehicle delivery routes through grocery stores. GTSP applications occur when only one vehicle is involved, as in the Covering Tour Problem (CTP) recently analyzed by Gendreau, Laporte and Semet. The CTP is defined on a graph H = (V U W, E), where E = {(vi, vj): i < j, vi, vj E V U W}, and the distance matrix C is defined on E. The aim is to construct a least length tour through a subset of V in such a way that each vertex of W is within a distance c of a vertex of the tour. Letting Vk = {vi E V: Cik < c}, for each vk E W, the CTP can then be stated as a GTSP on the subgraph of G of H induced by $U_{v_k \in w}$ Vk. Examples of the CTP include post-box location problems" in which W is a set of population centroids (vk E W would represent, for example, all people living in city block k), Vk is a set of potential post-box locations within walking distance c from Vk, and the objective is to design a minimum length tour for a postal vehicle visiting the selected locations on a regular basis. The CTP is sometimes referred to as the Travelling Circus Problem as it is the problem faced by circus managers wishing to tour at minimum cost a set of villages, while ensuring that any village not on the tour is within c distance limits from it.

ACO products offer excellence in all areas of stormwater and cable management. Systems are purposed designed for high-trafficked areas where safety comes first.
Common applications include:

- Road and highways, including bridges and tunnels
- Mass transit systems: bus, rail and light rail
- Seaports
- Airports

This kind of algorithms can be use **in Morocco** at this time in the establishment of the railway between Marrakech and Agadir since the government start studies just these days. And it could be used in the distribution companies such as AMANA or CARRY to serve as many points as possible in much faster way

Also, the future Nigeria - Morocco pipeline project, proposal of the optimal routing based on the **Dijkstra Algorithm**

Conclusions:

As a civil engineering student, I think optimization project would be more efficient if they were carried out by companies because they are in direct relation with the real things.

Optimization is very important to achieve certain goals in a n efficient way, and to not stress nature resources and to protect environment.

---------- References ----------

http://www.aco-metaheuristic.org/aco-code/

http://adibaba.github.io/ACOTSPJava/

https://github.com/Vampboy/Ant-Colony-Optimization

https://courses.cs.washington.edu/courses/cse373/17su/lectures/Lecture%2016%20-%20Dijkstra%27s%20Algorithm.pdf

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Applications

https://en.wikipedia.org/wiki/Mathematical_optimization

https://www.acnusa.fr/fr/la-pollution-de-lair/sources/18

https://www.researchgate.net/publication/334127641_Site_location_and_Pipeline_routing_optimization_for_Future_natural_gas_importing_terminal_project_in_Morocco